Designing an Agent System for Controlling a Robotic Soccer Team †

Alejandro J. García Gerardo I. Simari Telma Delladio

Grupo de Robótica Cognitiva
Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur. Av. Alem 1253, (8000) Bahía Blanca, Argentina
Tel: ++54 291 459 5135 - Fax: ++54 291 459 5136
{ajg,gis,td}@cs.uns.edu.ar

Abstract

Robotic soccer is a way of putting different developments in intelligent agents into practice, including not only problems such as multi-agent planning and coordination, but also physical problems related to vision and communication subsystems. Because these problems cannot be all taken into account beforehand, the system must be designed to be robust enough to recover from any eventualities.

In this work, we present the design used as the basis for an agents system implemented for the control of a team of robots for the E-League competition in RoboCup 2004. The implementation of the system was carried out following a layered design, with the objective of having a set of Service Layers, each of which is associated with a different level of abstraction. This layered design allows to construct a functional system with basic services that can be tested and refined progressively. The layers that are proposed as a basis for the arquitecture of a robotic soccer team offer a modular design, allowing the possibility of reuse in other robotic soccer leagues.

Finally, the agents are implemented using the PROLOG language; the three uppermost layers in the hierarchy offer interfaces designed explicitly for this language.

Keywords: Intelligent Agents, Multi-agent Systems, Cognitive Robotics, Logic Programming.

1 Problem Description and Background

The development and implementation of any type of computational system requires an adequate analysis of requirements that allows the definition of the system that will be modelled, restrictions imposed (be them on requirements or on the specification), and performance parameters by which the behavior of the system can be evaluated.

†Workshop de agentes y sistemas inteligentes (WASI)

In diverse research fields, systems are not always developed as a solution of a particular problem; sometimes, they are conceived in the form of testbeds for new theories, tools, and problem solving techniques. This methodology is common practice in the area of inteligent systems, in which problems with a rich structure are considered in order to address reasoning, belief revision, comunication, learning, and autonomy, among other aspects. In the last few years, the game of soccer has drawn much attention in the area of multi-agent system research and development [ASV96]. This is due to the fact that soccer is a complex and challenging domain that is useful in the evaluation of different kinds of developments that have been carried out in the field.

The game of soccer can be seen as a well defined system: the number and type of players, duration of play, allowed behaviors, and penalizations (among other aspects of the game) are governed by a well defined set of rules that are known to all participants. However, the interaction among the players cannot be defined beforehand. Each team is composed of players that must cooperate in order to reach their goal of winning the game. They must also take into account the existence of the opposing team, which also has the goal of winning the game.

Robotic soccer is a way of putting different developments in intelligent agents into practice. This includes developments in autonomous, cooperative, competitive, reasoning, learning, and revision systems [Wei98, HV]. Furthermore, it is useful in identifying problems related to aspects concerned with vision and communication. This type of problems cannot be all taken into account beforehand, and therefore demand that the system be designed in to be robust enough to recover from eventualities of this type. Robotic soccer is a complex domain, and it is necessary to take into account several aspects related to the robots. Each robot has sensors and effectors which are prone to failure. The enviroment is dynamic so there is no chance of knowing in advance the situations that can arise in a game. Therefore, it is necessary to be able to recover from adverse situations like sensorial or effectorial failure, and the decisions needed to carry out the recovery process have to be taken quickly.

In this work, we present the design used as the basis for an agents system implemented for the control of a team of robots for the E-League competition in RoboCup 2004 [LIS]. The implementation of the system was carried out following a layered design; the objective is to have a set of Service Layers, each of which is associated with a different level of abstraction. Each layer solves a different set of problems by means of the services that it offers; these services can then be used by the upper layers. This layered design allows to construct a functional system with basic services that can be tested and refined progressively. As the low level service layers are implemented, the upper ones can be designed and tested using prototypes. This is also useful in testing different situations that may arise. For example, a prototype of the layers that offer services of sensorial and effectorial information can be used to evaluate and refine the design of these layers which implement the robot's behavior. The layers that are proposed as a basis for the arquitecture of a robotic soccer team offer a modular design. A team designed following this scheme could then be modified easily and reused in another robotic soccer league.

2 The E-League

The E-League [COL] is currently a part of RoboCup [ROB]. It was conceived with the objective of initiating students in robotic research, and to provide an environment that is adequate for research and teaching [ABLS03]. This league provides common basic services to all of the participants, such as vision and communication. Teams can use low cost kits such as [LEG] and concentrate on the development and study of Artificial Intelligence techniques such as planning, multi-agent development, communication, and learning. The league's most important feature is its simple and modular structure. There are only three basic components that must be available to obtain a functional team: a vision module that works as the robot's perception component, a communication module that allows actions to be communicated to the robots, and a control module that is implemented by agents that control the robots on the field of play.

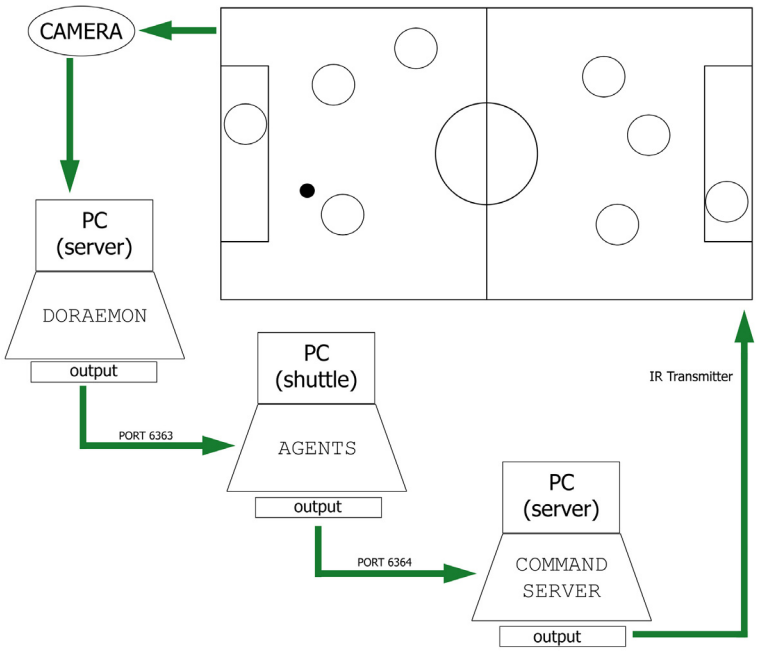

Figure 1: E-League setup (taken from [GSD+04])

Each team has one or more auxiliary computers in which the agents are executed. These agents communicate with the vision component in order to obtain information about what happens on the field, and send messages to the robots by means of a communication module (see Figure 1). Even though the league does not define a standard platform for the construction of the robots, it does impose restrictions over the processing and memory capacity. This allows the use of low cost robotic kits, many of which fall under these restrictions. The system we developed was implemented using Lego Mindstorms kits [LEG], which are within the rules of the league.

Each team is composed of four robots. One of the robots can act as goalkeeper, but this is not strictly necessary. There are restrictions over the size of the robots, their shape, and the

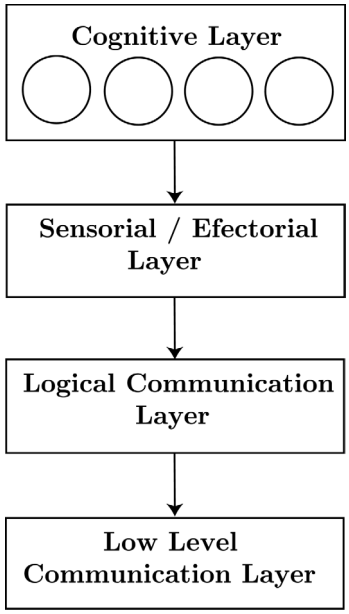

Figure 2: A proposed architecture for the implementation of an E-League team.

components used in their construction. Even though the robots do not communicate amongst themselves, the processes that control them can do so.

3 Design of the agent system

The proposed design considers the construction of the system based on services that are associated with four main layers (see Figure 2), each of which covers different levels of abstraction of the problem to be solved. Each of these layers provides services to the upper layers, which are implemented using those provided by the lower layers. This layered design allows local modifications. The implementation of some services can be modified without provoking many changes in other layers using these services.

3.1 Low level communication layer

We associate with this layer the basic hardware and software support that is provided by the league. This includes physical support, such as infrared transmitters, video camera, communication network, and common software. The software that is provided by the league includes video and command communication servers. Furthermore, this includes the communication processes by which the referee conveys his decisions to both teams.

The video server, called *Doraemon* [JBA, AB], is part of the software support provided by the league. A video camera covers the field of play and this server processes the images that it obtains, generating information packets that are then made available to be used by the agents that control the both teams' robots. The packets that are generated by the video server

provide information about the objects that we defined above and that are on the field of play. Information is transmitted about the position, orientation, and speed of these objects. This vision software allows the definition of different types of objects [AB]; however, during game play, the only objects that it recognizes are the ball and the players (robots).

Communication between the teams and the vision software follows the client-server model. The process or processes that implement each soccer team communicate through the data network to a UDP socket over which the vision server sends the data packets that are transmitted in ASCII format. The parameters that are necessary for establishing this communication are determined at configuration time. The information that is generated by the video server is transmitted to all of its clients, and has the following format:

0	No. obj.	Frame no.	Time diff.						
1	CamX	CamY	CamZ						
2	Type	Name	Found?	PosX	PosY	PosZ	Orient.	VelX	VelY
⋮	...	...	...	...	...	...	...	...	...
10	Type	Name	Found?	PosX	PosY	PosZ	Orient.	VelX	VelY

Line 0 indicates the number of objects that are being tracked, a packet number, and the time that has elapsed between the last two transmissions. Line 1 contains the (x,y,z) coordinates of the camera with respect to the field of play. The following nine lines contain the information about each of the robots (for both teams) and the ball. Each of these lines contains the following information:

1. Object type: indicates if the object is a ball, robot, or other type of object.
2. Object name: an identifier for the object; if it's the ball, the identifier could be *ball* and, for the robots, the name will be the identifier associated to each of them when the server was configured.
3. A flag indicating if the object was effectively identified by the vision server. If this does not occur, it indicates that the object was not found (NoFnd) and the information associated with it is only a prediction carried out by the vision software.
4. The object's (x,y,z) coordinates.
5. The object's orientation expressed in radians. The ball does not have an orientation.
6. The object's speed.

The league also provides a command communication software called *Command Server* (abbreviated CS from now on), which allows the agents, who control the robots, to send messages to the field. As we have mentioned, the processing and memory capabilities of the robots is limited, and the control software must therefore reside and execute on auxiliary computers. In this way, the decision processes are carried out by these agents, and the decisions are then

communicated to the robots through the CS. This communication process also follows the client-server model. As with the vision server, the necessary parameters (host, ports, etc) are determined during the configuration stage.

As we have seen, the CS is the bridge between the agent processes and the robots. The CS listens over a socket for the messages from its clients (agents), which are in ASCII format and have the following form:

```
[name]:[msg]\n
```

where `[name]` if the identifier of the robot to which the message is directed, and `[msg]` is the message to be sent. The CS maintains an 8 byte buffer (one byte for each player) that it uses to store the messages sent to the robots. Every time it receives a message it updates the byte corresponding to the robot to which it is directed. This information is transmitted to the field through an infrared transmitter connected to a standard serial or USB port. Each of the messages that are transmitted is a 10 byte packet of the form:

```
[START][command-buffer][CHKSUM]
```

where `[START]` is an initial byte, `[CHKSUM]` is a control byte, and `[command-buffer]` is the information that, at the moment, was stored in the buffer maintained by the CS. This information contains the 8 bytes that represent the messages for each of the players on the field. A special firmware [BRI] used in the robots allows them to identify the byte that contains the message directed to each robot in particular within the packet sent by the CS.

The flow of the game is controlled externally by a referee, whose decisions are communicated to the robots through a predetermined interface. Therefore, because the referee is part of the environment, the agents must be designed to recognize his decisions and carry out the necessary actions for the robots to respond accordingly. For this reason, the system that we implemented has a process by which it continuosly senses the referee's decisions and, depending on them, updates the state of the game. In Figure 3, we show part of the state space for a game. Normal game play is represented by *"Game on"*; if the referee blows the whistle, the game is stopped and its state can change to one of several options, depending on the referee's decision. For example, if the referee calls a penalty kick for the opposing team, the current state will change to *"Waiting their penalty"*.

3.2 Logical Communication Layer

This layer offers as its services a set of routines that are the interface between the low level communication layer and the upper layers. Basically, the necessary services that allow communication with the vision and communication servers are provided.

The routines that implement these services have been developed in the C programming language and, using these routines, PROLOG predicates are implemented. These predicates are

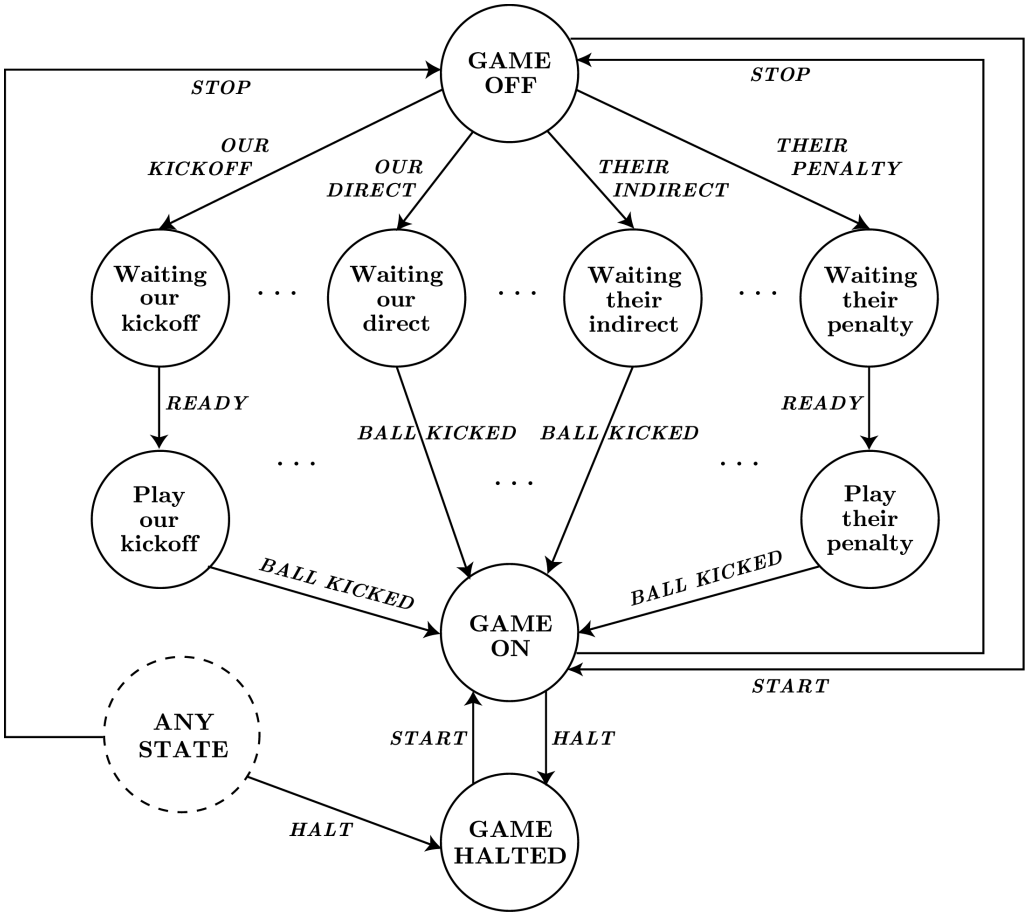

Figure 3: Part of the game's state space and the transitions caused by the referee's decisions.

used by the upper layers in order to communicate with the different servers. This decision has several advantages. One of these advantages has to do with the execution of actions. As we mentioned before, the environment is highly dynamic, which causes the states of the world to change quickly. Therefore, it is necessary for the robots to be able to react accordingly to this dynamism. Moreover, the information obtained by the robots can be wrong due to sensorial failure; after recovering from such a failure, the current situation could be completly different from the one previously perceived. In this type of cases, the system has to be able to analyze new situations, and quickly decide which actions should be taken by the robots. For these reasons some critical services, like the ones which sense the environment, have been implemented using the C programming language. Nevertheless, many of the aspects related to the design of the behavior of a team require richer languages that can more easily reflect the decision processes made by the agents. For this reason, we chose the PROLOG language for the implementation of the agents' behavior. Furthermore, the majority of the developments carried out in our research group are implemented in PROLOG.

One of the routines implemented in C reads the packets generated by the video server, and returns the information in an adequate format. This routine is used in the implementation of a PROLOG query predicate (called `dora/9`) which has parameters that are used for returning

the information that is obtained. This PROLOG predicate is used by the upper layers in the implementation of the aforementioned agents.

Another routine establishes the communication with the CS. This routine accepts the identifier of one of the robots and the mesage to be sent, builds a packet with this information, and delivers it to the CS so that it can send it via the infrared transmitters to the field. With this routine, a PROLOG predicate (called `msg2robot/2`) is implemented, which will later be used by the agents to send commands (actions) to the robots. Each robot has a set of basic actions that are implemented and executed by the robot's own processor. These are basic movement actions, such as turning the motors on and off.

Finally, a routine that continuously senses the referee decision is implemented as the basis of a PROLOG predicate (called `referee/1`) which allows to query the current state of the game. This predicate is used by the agents to carry out their decision processes.

The existence of this layer allow us to disregard the physical structure of the environment in which the team of robots is embedded. For example, it is possible to implement the services of this layer based on a simulated environment like the *RoboCup Simulated League*. If the interface of the services offered by this layer remain unchanged, then the rest of the upper layers can also remain unchanged.

3.3 Sensorial/Effectorial Layer

In this layer, the visual information is processed and translated into terms that express states of the world. The coordinates and speeds of the robots and the ball can be interpreted to express particular situations. Query and action predicates related to particular game situations are defined. Among others, we define PROLOG predicates to determine:

- The robots' and the ball's positions,
- player and/or team that is closest to the ball,
- distances between different objects on the field, between players, rival players, etc, and
- plays involving high levels of risk.

All of these predicates are implemented by querying and analyzing the information from the video server. Then, the situations modelled through these predicates are used in the upper layer to model and implement the team's game strategy.

Furthermore, predicates are also designed to implement the actions that must be carried out by the robots. These are primitive actions that can later be combined, in the upper layer, to build a more complex plan of action. General primitive movement actions are defined, such as rotation and forward and backward movements. Moreover, there are primitive actions that implement basic actions that every agent implemented in the upper layer can use. Some examples of these types of actions are: go to a given position, kick, move to a given position while avoiding obstacles, among others.

3.4 Cognitive Layer

This layer is responsible for the design of the agents that implement the team. Like in any other agent system, the agents perceive information about the environment and reason about the actions to take, as we show in Figure 4. This layer is formed by a set of logical agents implemented using logic programming (PROLOG). Each agent perceives the information from the environment (composed of the vision information and the state of the game) and acts in consequence to reach its goals according to its design. In order to do this, each agent is modelled

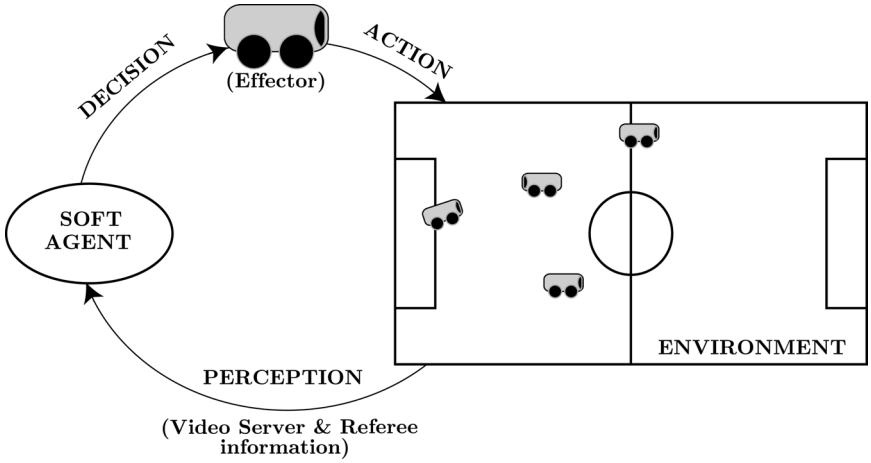

Figure 4: Perception and action loop in our agent-based system.

through a perception/action loop of the form:

```
agent ←
      referee(GameState),
      dora(PerceptionInfoList),
      decide(GameState, PerceptionInfoList, Msg),
      msg2robot(Msg),
      agent.
```

Each agent perceives the state of the game through the referee's decisions, obtains the information of the field of play through the video server and, according to this data, chooses the actions to perform. These actions are sent to the robot via de CS; it should be noted that, from the point of view of this layer, the robots are essentially complex effectors. For example, for the goalkeeper, the following situation may arise: the game state is normal (the referee has not made a call), the ball is in posession of the opponent team and close to the goal. In this case, the goalkeeper will position itself in the part of the goal that will allow it to intercept an eventual shot at the goal.

4 Conclusions and Future Work

In this work, we have presented an agents system for controlling a robotic soccer team based on the organization into different Service Layers. This design allows the abstraction and modularization of the different aspects of the complex domain that robotic soccer represents. The design of the team of agents was carried out using a logic-based model. Logic programming is a useful tool for the implementation of reasoning systems that offer the possibility of rapid prototyping of agents and declarative design. As we have seen, this was taken into account in the design of the different layers, which provide a basis for the development of agents implemented using PROLOG.

Having built a functional team of agents using this model to participate in the RoboCup competitions, we are currently working on the development of more complex agents that incorporate a wide range of AI techniques developed within LIDIA from the fields of planning, argumentation, learning, belief revision, and game theory and decision theory, among others.

Acknowledgements

We are very grateful to the *Association of Computing Machinery* (ACM) for granting us a scholarship that allowed us to cover most of the costs of participating this year in the competition in Lisbon, and to Dr. Elizabeth Sklar and Dr. Simon Parsons of the University of Columbia in New York, for making this possible. We are also grateful to the Department of Computer Science and Engineering, Universidad Nacional del Sur, for making its space and resources available to us at any time. We would also like to thank the rest of our supporters, which can be found at `http://cs.uns.edu.ar/~ajg/matebots/`. The project is also partially supported by the *Agencia Nacional de Promoción Científica y Tecnológica* (PICT 13096), and the Secretaría de Ciencia y Tecnología of Universidad Nacional del Sur. LIDIA is a Member of the IICyTI (Instituto de Investigación en Ciencia y Tecnología Informática). Gerardo Simari is partially supported by *Comisión de Investigaciones Científicas*, (CIC) Gobierno de la Provincia de Buenos Aires, and Telma Delladio is partially supported by *Consejo Nacional de Investigaciones Científicas y Técnicas* (CONICET).

Finally, this work describes part of the work that has been carried out in the Cognitive Robotics group as a general effort to participate in the RoboCup competitions. Apart from the authors, the following people also take part in this project: Diego García, Mariano Tucat, Fernando Martín, Sebastián Gottifredi, and Nicolás Rotstein.

References

[AB] John Anderson and Jacky Baltes. Doraemon user's manual. http://sourceforge.net/projects/robocup-video.

[ABLS03] John Anderson, Jacky Baltes, David Livingston, and Elizabeth Sklar. Toward an undergraduate league for robocup. In *Proceedings of the RoboCup Symposium*, 2003.

[ASV96] S. Achim, P. Stone, and M. Veloso. Building a dedicated robotic soccer system, 1996. In Proceedings of the IROS-96 Workshop on RoboCup.

[BRI] http://brickos.sourceforge.net/. First open-source operating system for the Lego Mindstorms RCX Controller.

[COL] http://agents.cs.columbia.edu/eleague/. Official E-League webpage.

[GSD+04] Alejandro J. García, Gerardo I. Simari, Telma Delladio, Diego R. García, Mariano Tucat, Nicolás D. Rotstein, Fernando A. Martín, and Sebastián Gottifredi. Cognitive robotics in a soccer game domain: A proposal for the e-league competition. In Universidad Nacional del Comahue, editor, *6to Workshop de Investigadores en Ciencias de la Computación (WICC)*, pages 289–293, 2004.

[HV] Kwun Han and Manuela Veloso. Automated robot behavior recognition applied to robotic soccer. Robotics Research: the Ninth International Symposium, pages 199–204. Springer-Verlag, London, 2000. Also in the Proceedings of IJCAI-99 Workshop on Team Behaviors and Plan Recognition.

[JBA] Benn Vosseteig Jacky Baltes and John Anderson. Robocup e-league video server. http://sourceforge.net/projects/robocup-video.

[LEG] http://www.legomindstorms.com. Lego Mindstorms robots and RCX controllers.

[LIS] http://www.robocup2004.pt. Official webpage of the RoboCup 2004 competition, held in Lisbon, Portugal.

[MAT] http://cs.uns.edu.ar/∼gis/robocup-tdp.htm. English version of the official Matebots E-League team webpage.

[ROB] http://www.robocup.org. Official webpage of the RoboCup Federation.

[Wei98] Gerhard Weiss. Learning to coordinate actions in multi-agent systems. In *Reading in Agents*, pages 481–486. Morgan Kaufmann, 1998.